AD-A238 252

NAVSWC TR 90-418

INVARIANT TARGET RECOGNITION USING SELF-ORGANIZING NETWORKS

BY BETH FARRAR   ALI FARSAIE   J. JOSEPH FULLER

WEAPONS SYSTEMS DEPARTMENT

OCTOBER 1990

NAVAL SURFACE WARFARE CENTER

Dahlgren, Virginia 22448-5000 • Silver Spring, Maryland 20903-5000

91-04021

NAVSWC TR 90-418

INVARIANT TARGET RECOGNITION USING SELF-ORGANIZING NETWORKS

BY BETH FARRAR   ALI FARSAIE   J. JOSEPH FULLER
WEAPONS SYSTEMS DEPARTMENT

OCTOBER 1990

Approved for public release; distribution is unlimited

NAVAL SURFACE WARFARE CENTER
Dahlgren, Virginia 22448-5000 • Silver Spring, Maryland 20903-5000

EXECUTIVE SUMMARY

This paper presents an overview of neural network technology and specifics of the self-organizing networks. An artificial neural network is described which recognizes objects regardless of their spatial orientation. This network is invariant to rotation, scale, and translation.

Invariance is built into the network by introducing a unique set of features which were developed in-house. This overcame the two shortcomings of long training times and combinatorial explosion of terms often present in other networks. Preliminary results suggest that with further refinement and enhancement, the system described in this report will have the capability to reliably recognize targets under adverse environmental conditions.

FOREWORD

An overview of Artificial Neural Networks technology and its application to target recognition is presented. Architecture and attributes of self-organizing networks are discussed in detail. Target recognition using self-organizing neural networks was investigated. The networks were trained to recognize targets regardless of the viewing position. Performance as a function of orientation angle, and scale were characterized as well as overall performance.

This technical report was reviewed by Frank Rucky, Head of the Concepts and Technology Branch and Kurt Mueller, Acting Head of the Advanced Weapons Division.

Approved by:

O. P. Credle

O. P. CREDLE, Head
Weapons Systems Department

CONTENTS

ILLUSTRATIONS

TABLES

INTRODUCTION

Building machines capable of human-like thinking and reasoning will probably require technology derived from the study of artificial intelligence and neural networks. For many years researchers have been examining the functioning of the brain trying to simulate some of its unique capabilities by computer. Already, these networks can simulate learning some of the functions that humans are able to perform almost automatically such as image and speech recognition. Although progress has been made in these systems, researchers have only scratched the surface and more work needs to be done to make the systems more versatile and accurate.

Similar to our current understanding of the human brain, neural network models are composed of dense interconnections of simple, nonlinear computational elements operating in parallel. These systems have many advantages over sequential machines including the high computation rate provided by massive parallelism. Neural networks are also more robust than conventional computational techniques since degradation of the input signal or failure of a few nodes or connections will not significantly impair overall performance. Neural networks show the most potential in areas where multiple hypotheses are pursued in parallel, high computation rates are required, and the current systems performance is far from equaling human performance.[1]

The goal of this report is to describe a system for automatic target recognition. The system is able to recognize a class of targets regardless of its angle of orientation, its relative size, and its position in the field of view. These factors, known as rotation, scaling, and translation, cause distortions in the object which make it look very different to the computer than the same image in its standard position. Also, noise in the picture (such as a cloud covering part of the target) can confuse the system into thinking that the noise is part of the image and thus make it more difficult for the automatic target recognition system to identify the target. Current target recognition systems are not able to completely overcome the noise problem. However, neural networks offer hope of achieving a solution to this problem.

BACKGROUND

AUTOMATIC TARGET RECOGNITION

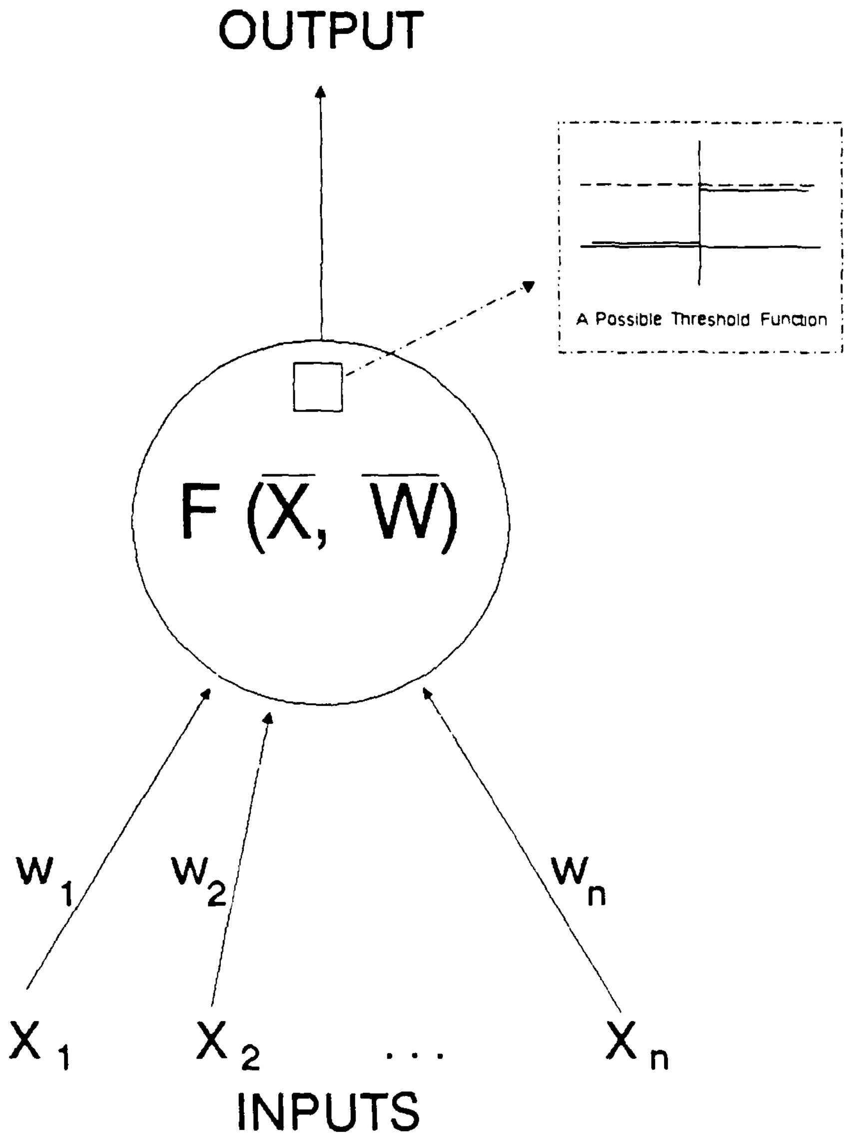
Target recognition can be defined as choosing the proper model structure to characterize a given target or, more simply, identifying a target as belonging to a certain class. Some criteria used for target identification are explained by Larimore[2] and Chellapa and Kashyap.[3] Researchers have attempted to identify unknown objects by comparing their 2-D silhouettes to silhouette models in a library. Edges composed of straight line segments have been used as features by many researchers for identification and satisfactory results have been obtained from these methods on conventional computers.[4,5]

Artificial neural networks have been trained to distinguish among digitized images of several targets presented in fixed positions with a high degree of success[6]. The success rate is much lower, however, when the targets are rotated, translated, or scaled. Previous techniques used to compensate for the rotation, translation, and scaling have included training the network on the images presented at different angles and designing networks so that the synaptic weights have affine invariance incorporated into their calculation.[7,8] These techniques were of limited success.[6]

ARTIFICIAL NEURAL SYSTEMS

Although they may seem big and extremely complex, in reality, artificial neural networks are simple elements which once interconnected form a powerful computational system. The fundamental elements of any neural network are the neurons and connection weights. The neuron is a simple computational element, usually a summer and threshholder, which is connected to other neurons via weighted connections. The connection weights specify the amount information is modified as it is passed between neurons. Various networks are formed by different connection schemes, neuronal functions, and training methods.

The neurons perform all of the mathematical functions of the network. A neuron receives input from the outside world, other neurons in the network, or, in some cases, both sources. The neurons use these inputs and the weights from its connections to other neurons to compute one output which it sends to other neurons in the network or the outside world. One of the differences between the various network architectures is the functions that the neurons perform. Some neurons also contain a threshold function which insures that the output is within the allowed range of values. Figure 1 shows a typical neuron.



$F(\bar{X}, \bar{W})$ is the function performed where
$\bar{X}$ is the vector of inputs $x_1, x_2, \ldots, x_n$ and
$\bar{W}$ is the vector of weights $w_1, w_2, \ldots, w_n$

FIGURE 1. A NEURON

For every neuron in the network there is a weight associated with each of its inputs. Each input is multiplied by its corresponding weight to determine its contribution to the output. These weights can be either excitatory or inhibitory. Excitatory weights increase the effect of the input value while inhibitory weights decrease the input's effect.

Neurons are generally grouped together in layers. Networks can be either one layer or multiple layers. Each neuron in one layer receives input from every node in the previous (source) layer, but each input is modified by the weight of the connections between the nodes. The output of each neuron in a layer, modified by the connection weight, is input to every node in the next (destination) layer. Although neurons can have different functions, neurons within a layer generally have the same function.

Learning takes place by modifying the values of the connection weights so that the actual output of the network gets closer to the desired output. Training, the process by which the network learns, can be performed using one of two different methods: supervised or unsupervised learning. In supervised learning, the actual outputs of the network are compared with the desired outputs (supplied by a teacher) to determine the network's error value. In contrast, during unsupervised learning, the network is not given a desired output; instead, the network uses its own method to determine the error value.* In either supervised or unsupervised learning, the network uses the error value to modify its weights. Since large changes in the weights would cause the network to become unstable, weight values are changed by only a small percentage of the error. Training time (measured in number of iterations, cycles through the network, or weight updates) depends upon the type of network used, the accuracy required for the given problem, and the difficulty of the problem. Training time is not necessarily critical because the real power comes from their speed and accuracy once they are trained.

Although there are many different types of networks, some are more well known than others. Experience with different models lead our effort toward the use of network structures similar to Kohonen's self-organizing feature maps.[9] This paper focuses on Kohonen's networks and describes some of the results from implementing variations of these networks for automatic target recognition.

---

*There are different methods which the network can use to compute error. One common method to compute the output of the network is to find the node which holds the smallest value. If this value is larger than a given threshold value, the difference between them is the error. Another method which is used in networks whose weights correspond to the input space, is to compare the value of the weight with the value of the corresponding input and use this difference as an error value.

KOHONEN'S SELF-ORGANIZING FEATURE MAPS

Kohonen developed a neural network which can function in either a supervised or unsupervised mode. The most well known version of Kohonen's network is the self-organizing feature maps, named for its ability to look like the input space. The supervised versions, learning vector quantization (LVQ) and improved learning vector quantization (LVQ2.1), are considered a special case of the original algorithm.[10]

SELF-ORGANIZING FEATURE MAPS

Kohonen's original network, self-organizing feature maps, consists of two layers; an input layer and an output layer. Each node in the input layer is connected to each node in the output layer as shown in Figure 2. Neurons in the output layer are arranged in a two dimensional grid with many local, within layer, connections. The input layer passes the data through the connections to the output layer which clusters the input vectors.

Each neuron in the output layer is connected to all of the nodes in its neighborhood. A neighborhood is defined as all of the surrounding nodes which are within a radius (R) of the center node. Initially R is fairly large and R shrinks with time until each node has only one node in its neighborhood (i.e. itself). Figure 3 shows the shrinkage of the neighborhood of the center node over a time period. One common approach is to initialize the radius R to half the number of nodes in each row of the output layer and let it decay linearly in time. However, no specific technique is proposed for decaying the radius R. Depending upon the problem, certain implementations may work better than others.

While the neighborhood is shrinking, the network is training. The training continues even after the neighborhood has shrunk to only one node. The error that is used to modify the weights of the network is defined as the difference between the input value and the corresponding weight value. In most cases, all of the weights in the network are not modified. Only the weights for the node that is closest to the input (the winning node) and the nodes in its neighborhood are modified. This method of modifying the weights has the effect of keeping nodes next to each other in the matrix responding to inputs that are fairly close to each other.

The first step in the algorithm for Kohonen's self-organizing feature maps is to initialize the network with random values. Usually these initial values are within the range of the input values. The

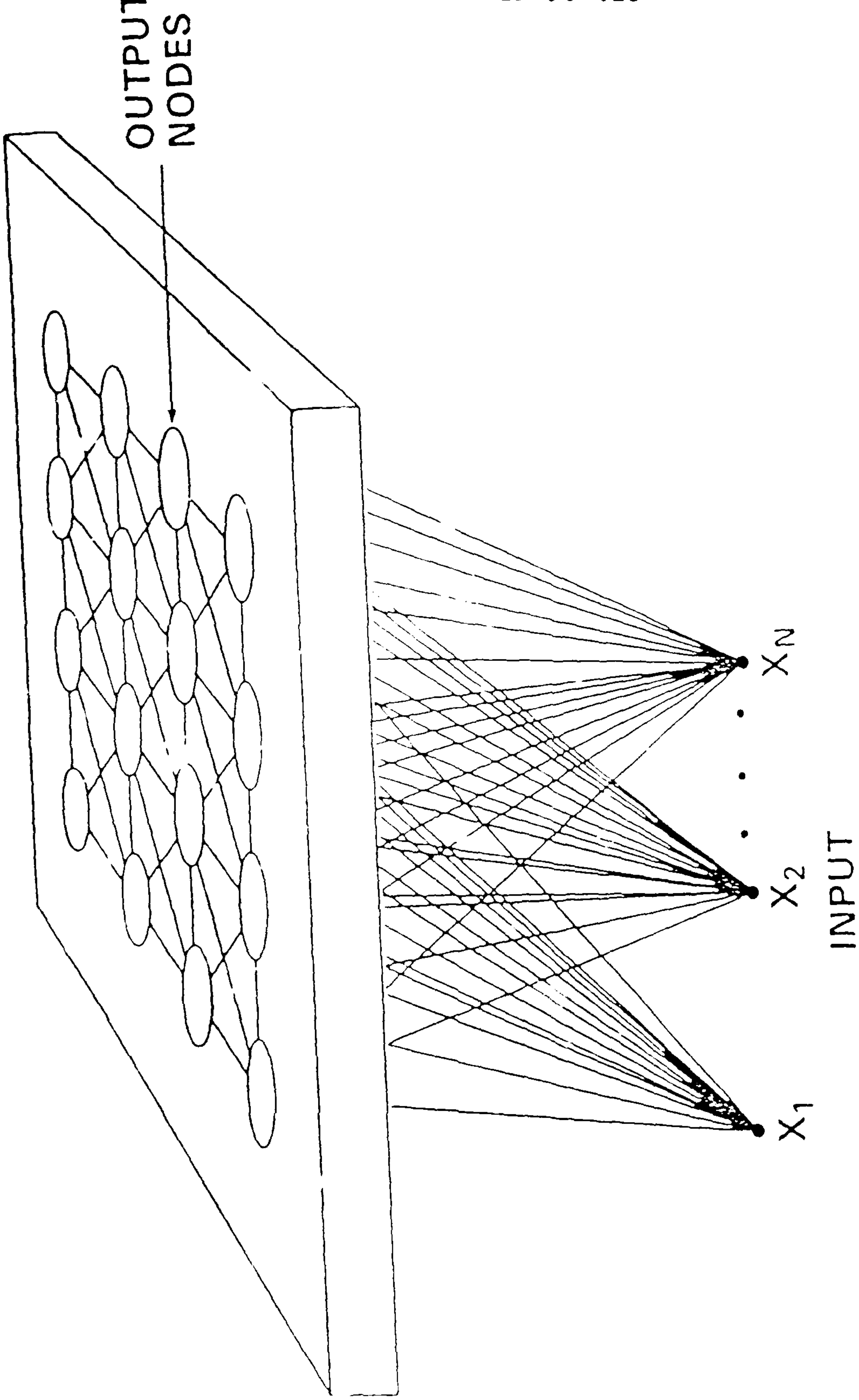


FIGURE 2. A SAMPLE KOHONEN NETWORK

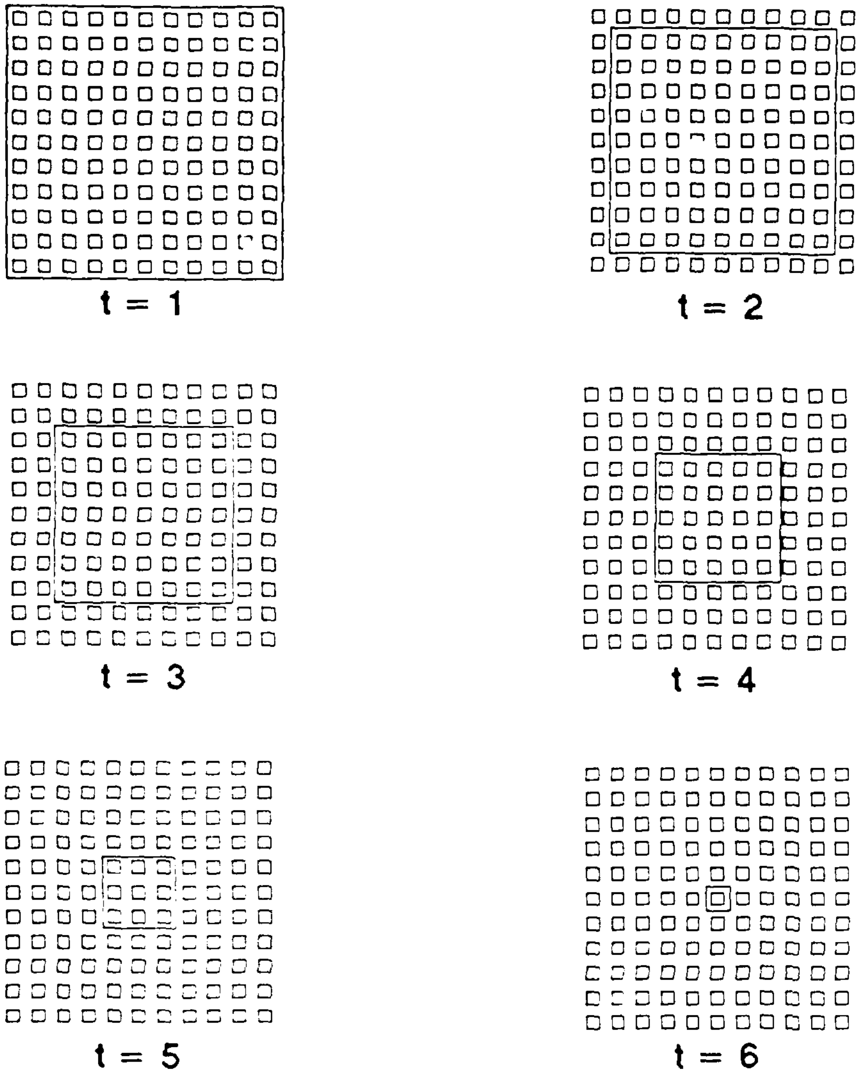


FIGURE 3. DECREASING SIZE OF NEIGHBORHOOD FOR THE CENTER NODE

network is then presented with input. With the presentation of new input, each neuron in the output layer computes the distance from its weights to the current input by:

$$d_j = \sum_{i=1}^{n} [x_i(t) - w_{ij}(t)]^2$$

where $x_i(t)$ is the input at time t, $w_{ij}(t)$ are the weights for node j at time t, and d_j is the distance to node j.

The node with the smallest distance (d_j) is considered the winner and is the only neuron that generates output. The weights in the winning node and its neighborhood are then moved slightly closer to the input by:

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)[x_i(t) - w_{ij}(t)]$$

where η is a learning parameter which starts out large (close to 1.0) at the beginning of training and gradually shrinks toward zero as the training progresses. Like the neighborhood shrinkage, the shrinkage of the factor η is left undetermined so that it can be adjusted to better fit the problem. A linear decay is suggested as a starting point for the shrinkage of η. Inputs are presented to the network and weights are modified for a given number of iterations or until some metrics are computed to indicate that the network has arrived at a steady state.

Due to the fact that the neighboring neurons in the output layer are pulled closer to the input with the winning neuron, the network begins to take on the shape of the input space. As it progresses during training more nodes will be pushed into areas with a higher concentration of input samples. If a picture was drawn of the network using the weights as coordinates and drawing lines between a neuron and the four neurons surrounding it (three in the case of nodes on the edges of the matrix) then none of the lines would cross each other and the picture would model the input space. The following examples illustrate this behavior of Kohonen's network.

Example 1: The Box Network

One fairly simple example of Kohonen's network which shows how the feature maps reflect the input space is the example of the box. This network consists of a ten by ten matrix of output nodes and two input nodes. The input to the network is a set of points randomly generated from a square area in the x-y plane. Initially the weights are set to small random values. After a few hundred iterations the weights begin to spread out evenly. If the nodes were plotted with the weights as small boxes and lines are drawn to the four neighboring nodes in the matrix, then the picture begins to look somewhat like a piece of graph paper as shown in Figure 4.

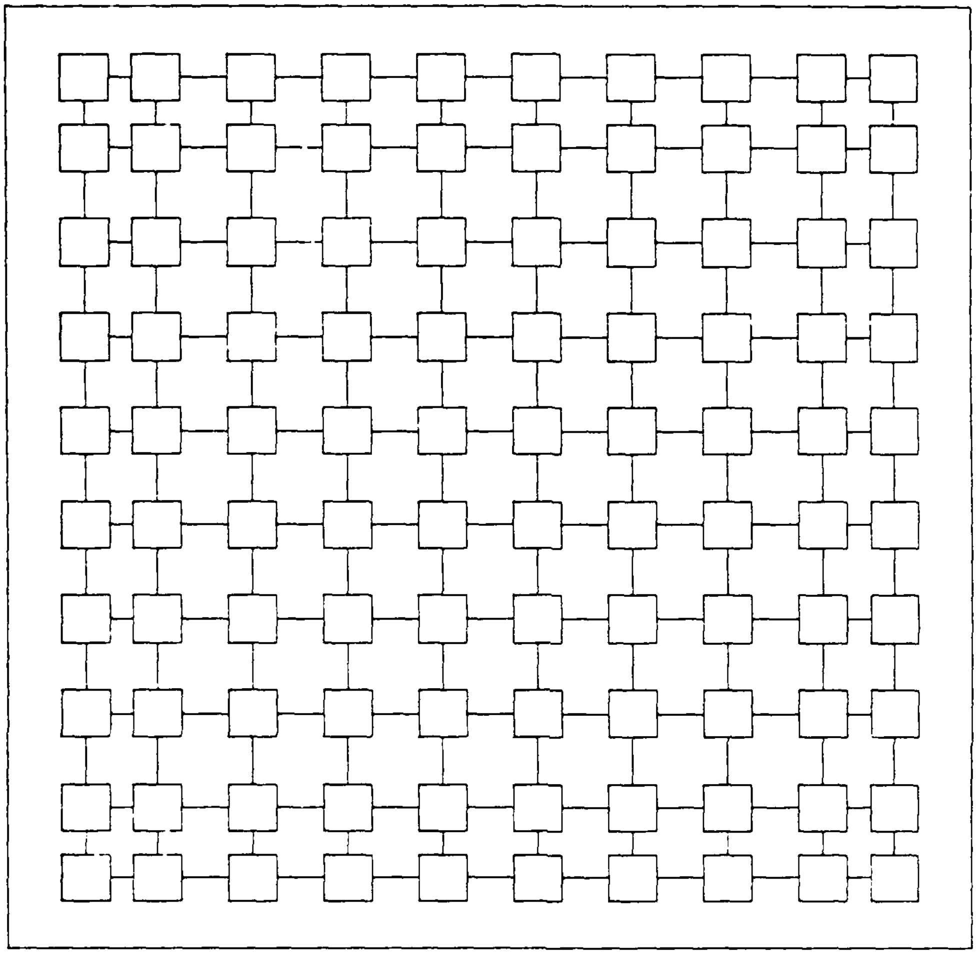


FIGURE 4. THE BOX NETWORK

A similar network has also been run with parts of the square region excluded from the input space. In this case the network placed very few neurons if any in the area that was excluded. These neurons are only placed there because they are pulled by the neurons on either side and are not able to respond to changes and move out of the excluded area. As an example, Figure 5 shows the weight distribution for the case in which no inputs are generated in the region $\{R(x, y) \mid x < 5, y < 5\}$.

Example 2: The Unit Circle Network

Another experimental example is the unit circle network. The network is initialized with the output layer consisting of a two by four matrix of nodes and the input layer with two nodes. The input consists of two points randomly chosen from the circumference of the unit circle with the origin placed at zero. The weights are initialized to random values along the circumference of the unit circle. The network was trained to divide the circumference of the circle into eight equal areas. The output is then drawn by connecting a dashed line from the origin of the graph to the point on the unit circle whose coordinates are the values of the weights See Figure 6.

This network was also run with a sixteen node output layer. The results were the same as in the above network except the circle was divided into sixteen different regions instead of eight. Experimentation with the network having different numbers of output nodes, it was determined that it would divide the circumference of the circle into as many equal regions as there were output nodes.

LEARNING VECTOR QUANTIZATION

Kohonen's learning vector quantization (LVQ) is a variation of the self-organizing feature maps network that uses supervised learning. This algorithm is frequently used to fine tune the weights after the completion of the self-organizing feature maps algorithm.

Using the LVQ algorithm, the desired output is known because the network has either been examined to determine the correct answers or been set up initially with the correct responses. An untrained network is initialized by setting the weights to actual values from one of the inputs in the region of interest for the node. Kohonen refers to these weight vectors as codebook vectors. Inputs are presented to the network and the distances are computed using the same function as in the self-organizing feature maps algorithm. If the desired output node is the winner then its weights are pushed slightly closer to the input by the equation:

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)[x_i(t) - w_{ij}(t)]$$

If the winning node is not the desired node then the weights in the winning node are pushed slightly away from the input by:

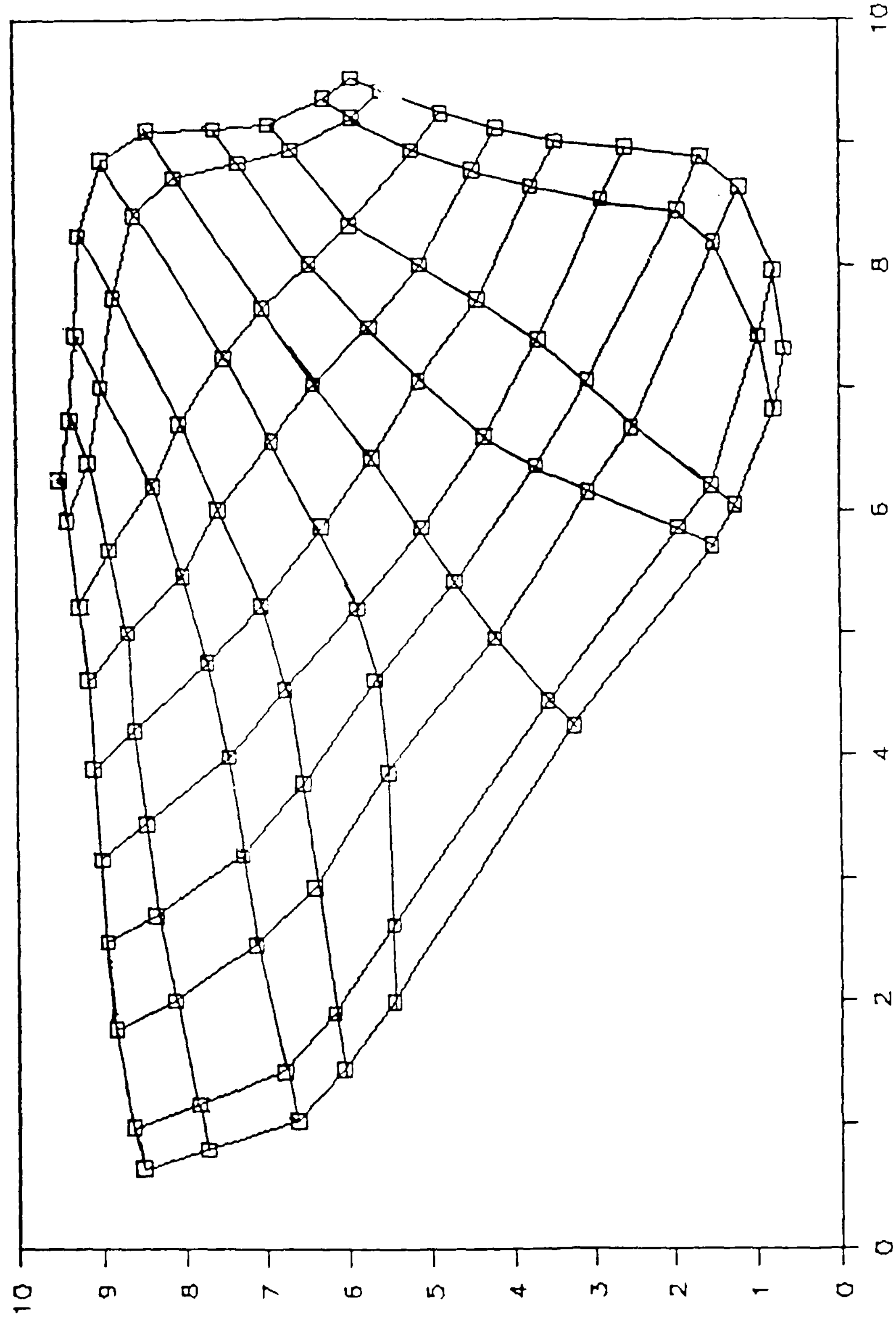
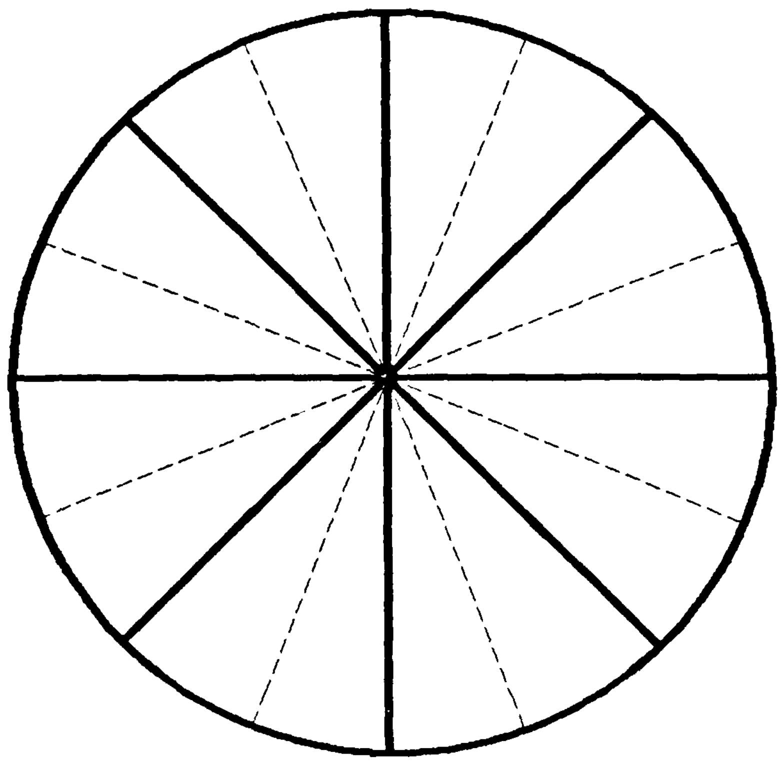


FIGURE 5. THE BOX NETWORK WITH NO POINTS GENERATED IN THE REGION $0 < X < 5$ AND $0 < Y < 5$



Dashed lines are the lines to the weight values. Solid lines show the regions of responsiveness for each node.

FIGURE 6. THE UNIT CIRCLE NETWORK

$$w_{ij}(t+1)=w_{ij}(t)-\eta(t)[x_i(t)-w_{ij}(t)]$$

where $x_i(t)$ is the input at time t, $w_{ij}(t)$ is the weight for node j at time t, and $\eta(t)$ is learning factor at time t. In this approach the weights are not only moved toward the correct responses but also moved away from the wrong responses. Because they are moved away from an incorrect response, the weights are fine tuned to more accurately portray the input.

IMPROVED LEARNING VECTOR QUANTIZATION

The improved learning vector quantization (LVQ2.1) is another variation of the supervised type Kohonen feature maps algorithm. This algorithm is the same as the algorithm for LVQ except for one change: there is a different action taken when the winning node is not the desired node. For LVQ2.1, not only is the winning node pushed away from the input but also the desired node is pushed toward the input using the following equations:

$$w_{ij}(t+1)=w_{ij}(t)-\eta(t)[x_i(t)-w_{ij}(t)]$$

$$w_{id}(t+1)=w_{id}(t)+\eta(t)[x_i(t)-w_{id}(t)]$$

where j refers to the winning node and d refers to the desired node.

The network that was selected for solving the automatic target recognition problem is the improved learning vector quantization network. This network performed better in tests than any other network that was tried. This paper presents a new approach to affine invariant target recognition using Kohonen's network.

TARGET RECOGNITION SYSTEM

DATA GENERATION

For the purpose of recognizing large image arrays of targets (100x100) without the combinatorial explosion of terms and connections that occur with a large image, an attempt was made to include information derived from correlations between pixels as described by Giles et.al. Furthermore, a neural network model was developed which accounts for aliasing and the jaggies which result from digitizing the picture on the computer screen.

The three targets shown in Figure 7 were used in this study. Each image is 100x100 pixels in size. The images shown are silhouette images. Thin, one pixel thick boundary, and thick, two pixel thick

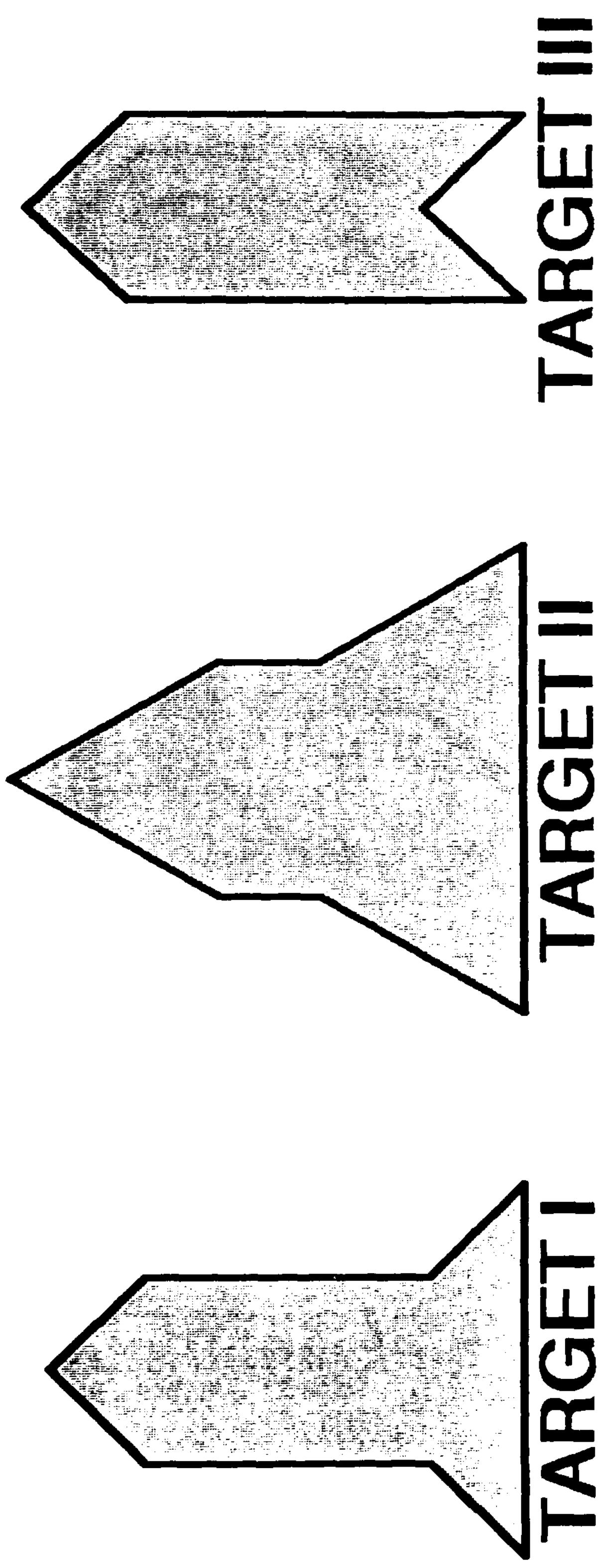


FIGURE 7. BENCHMARK TARGETS

boundary, edge images were also used. The targets shown in Figure 7 are in standard position; the targets were also rotated, scaled and translated in this project. No noise was added to the pictures in this stage of the project.

FEATURE EXTRACTION

A procedure for selection and extraction of features was developed to provide a set of feature vectors for use with the neural network model. Pixels in the image are considered either one or zero where one corresponds to a pixel in the target and zero corresponds to a background pixel. Components of the vector X represent the features extracted from an image, where X is defined as follows:

$X = \{x_1, x_2, x_3\}$

x_1 = the total number of pixels in the image that are not background.

x_2 = the sum of all of the products of pixels with the pixels which are the same distance from a designated origin but 90 degrees apart.

x_3 = the same as x_2 except for pixels 180 degrees apart.

At first glance, the above features appeared to be invariant - at least under translation and rotation - if the proper origin is selected. However, close inspection of these features revealed that aliasing may cause any or all of the features to fail to be invariant.

To remedy the problems described above, an average number of pixels representing a target was chosen rather than an absolute number of pixels. Deviation from this assumed average is a function of the rotation angle as well as the inherent problem of aliasing. Using this approach, a learning paradigm was selected which accounted for average behavior in a system. It appeared that although the features described above are not truly rotationally invariant when a transformation is applied to a digitized target, they are nearly rotationally invariant and a paradigm which accounted for fluctuations about an assumed mean would work well. This led to the choice of the LVQ2.1 algorithm.

After several training sessions in which an attempt was made to implement rotational invariance, it became apparent that aliasing and jaggies caused the features extracted at the mid angles (25 degrees to 65 degrees) to differ substantially from those extracted from the same target at angles near zero degree and 90 degrees. Therefore, it was decided to use two feature vectors to represent each target. The first vector was composed of features initialized with the target at standard position (at 90 degree angle), as shown in Figure 7, and the second vector used the same features initialized at mid angles (45 degrees).

Translation invariance was easily incorporated into the network by computing features with respect to the center of gravity of the target. Thus the designated origin referred to in the definitions of features x_2 and x_3 is the center of gravity of the target.

Features were adjusted for scale invariance by a simple ratio and proportion scheme. The radius (R) of the smallest circle centered at the target's center of gravity, which enclosed all of the target's pixels was found. Each of the features was then multiplied by $144/R^2$, where 144 is simply a base value indicating the square of the radius of the circle which encloses the standard size training targets.

TARGET RECOGNITION

First the features were extracted for the targets of interest, and then the feature vectors were used for training and testing. Kohonen's LVQ2.1 technique was used for recognition. Sample feature vectors for the targets were placed in the codebook vectors to initialize them.

A methodology was formulated and algorithms developed for training. A randomly selected target was rotated at an arbitrary angle between zero and 180 degrees while keeping target position fixed (no translation) and target size standard (i.e., scale factor of 1:1). Features were extracted and used to generate the feature vector as described before. Next, these feature vectors were introduced to the neural network model and the codebook vectors were computed and updated. During training the network assigned two codebook vectors to each target. This process was repeated for 1000 iterations, and resulted in the two codebook vectors moving toward the center of the cluster that represents the target. Iterations of 200, 400, and 600 were also considered during training processes.

During testing, one of the three targets was randomly selected, rotated, scaled, and translated. Feature vectors were extracted from the image and then presented to the network for recognition. These training and testing procedures were repeated for all of three sets of images (i.e., silhouette, thin edge, thick edge).

RESULTS

The recognition network was tested for many combinations of orientation angles, scale factors, and translations. Angles were varied from zero to 90 degrees, scale factors ranged from 0.49 to 1.7, and the translations ranged from -15 to 15 in both the x and y directions.

Performance of the network varied only 10 percent as a function of orientation angle when silhouette and thick edge images were used. In the case of thin edge images, the network perforce varied considerably and did especially poor at angles near 45 degrees. Figure 8 shows the percent correctly classified as a function of orientation angle.

It appeared that in the case of thin edge images the network did well for scale factors from 0.81 to 1.2, but the performance fell off sharply as the scale factor increased or decreased beyond this range. When thick edge images were used the network performed very well for scale factors from 0.81 to 1.44 with a correct recognition rate of 97

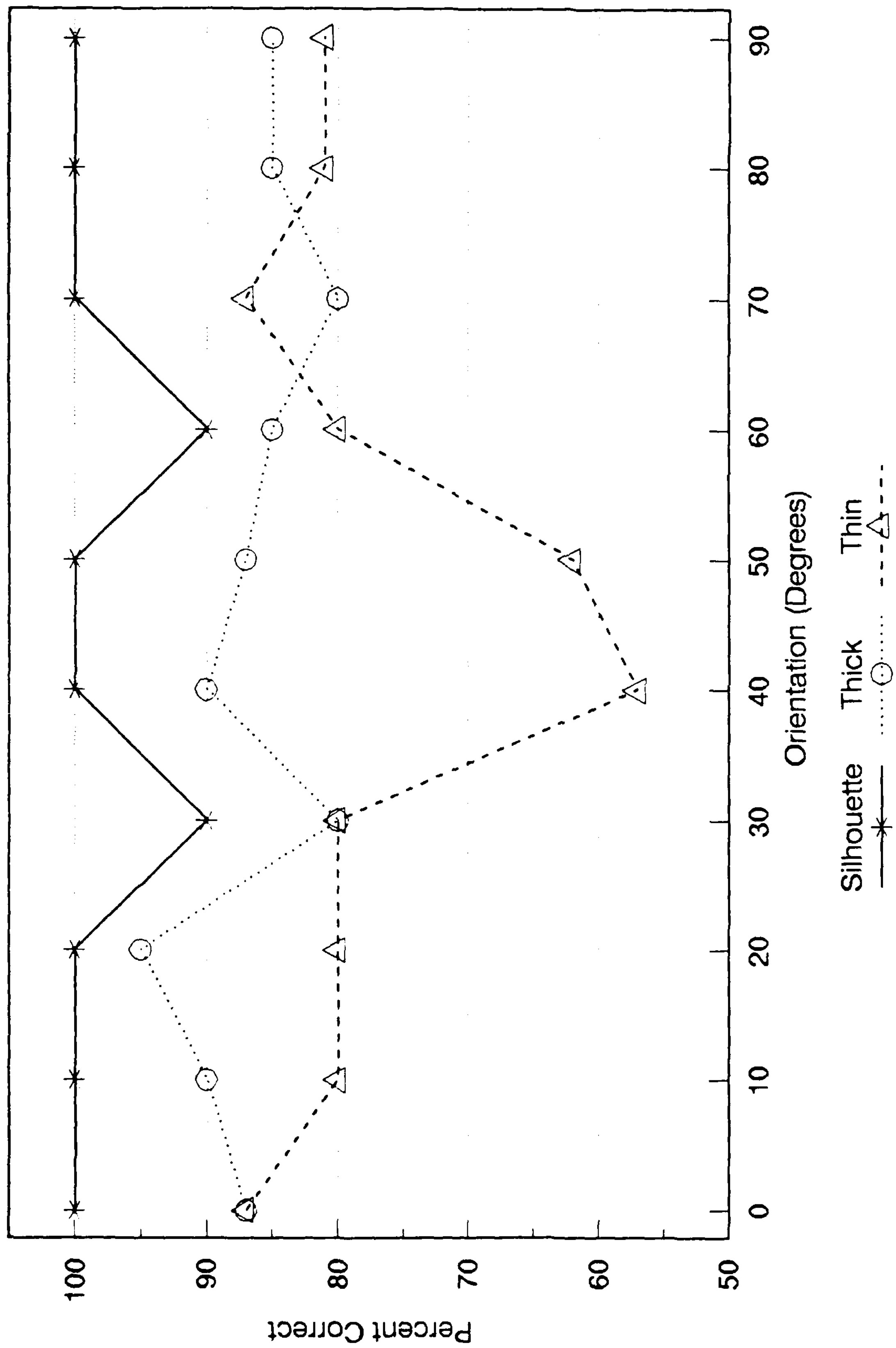


FIGURE 8. NETWORK PERFORMANCE AS A FUNCTION OF ORIENTATION

percent. In the case of silhouette images recognition rates as high as 100 percent were achieved, and where the target size was reduced the accuracy dropped by three percent. Figure 9 shows the percent correctly classified as a function of scale factor.

As expected, the network trained on silhouette images exhibited the best overall performance, achieving an accuracy of 99 percent when trained for 1000 iterations (Table 1). In general thick edge images were superior to thin edge images for identifying targets (90.0 vs. 70.0 percent). As Table 1 also shows, no appreciable improvement was obtained by increasing the number of iterations for either the thick edge or thin edge images.

CONCLUSION AND SUGGESTION FOR FURTHER STUDY

The test results clearly demonstrated that the feature extraction technique described here, along with LVQ2.1 algorithm, represents an efficient and effective method for target recognition which is invariant under any of the affine transformations of scaling, translating, and rotating. Using only two codebook vectors for each target and only three features per vector, a high rate of success was achieved in distinguishing among three targets presented at varying spatial orientations. In addition to the rapid training process and efficient identification, a new set of features was introduced which may be easily generalized to account for more complex structured targets.

Work is still underway on the problem of identifying targets by feature extraction with a Kohonen type network. Emphasis is being placed on establishing criteria for determining adequate training time, incorporation of statistical measures to determine probabilities of detection, and assigning measures of confidence to the response of the network. Other areas related to automatic target recognition which are currently being studied include developing more efficient scale invariant algorithms, and perhaps more significantly, designing networks to work in the presence of a low signal to noise ratio.

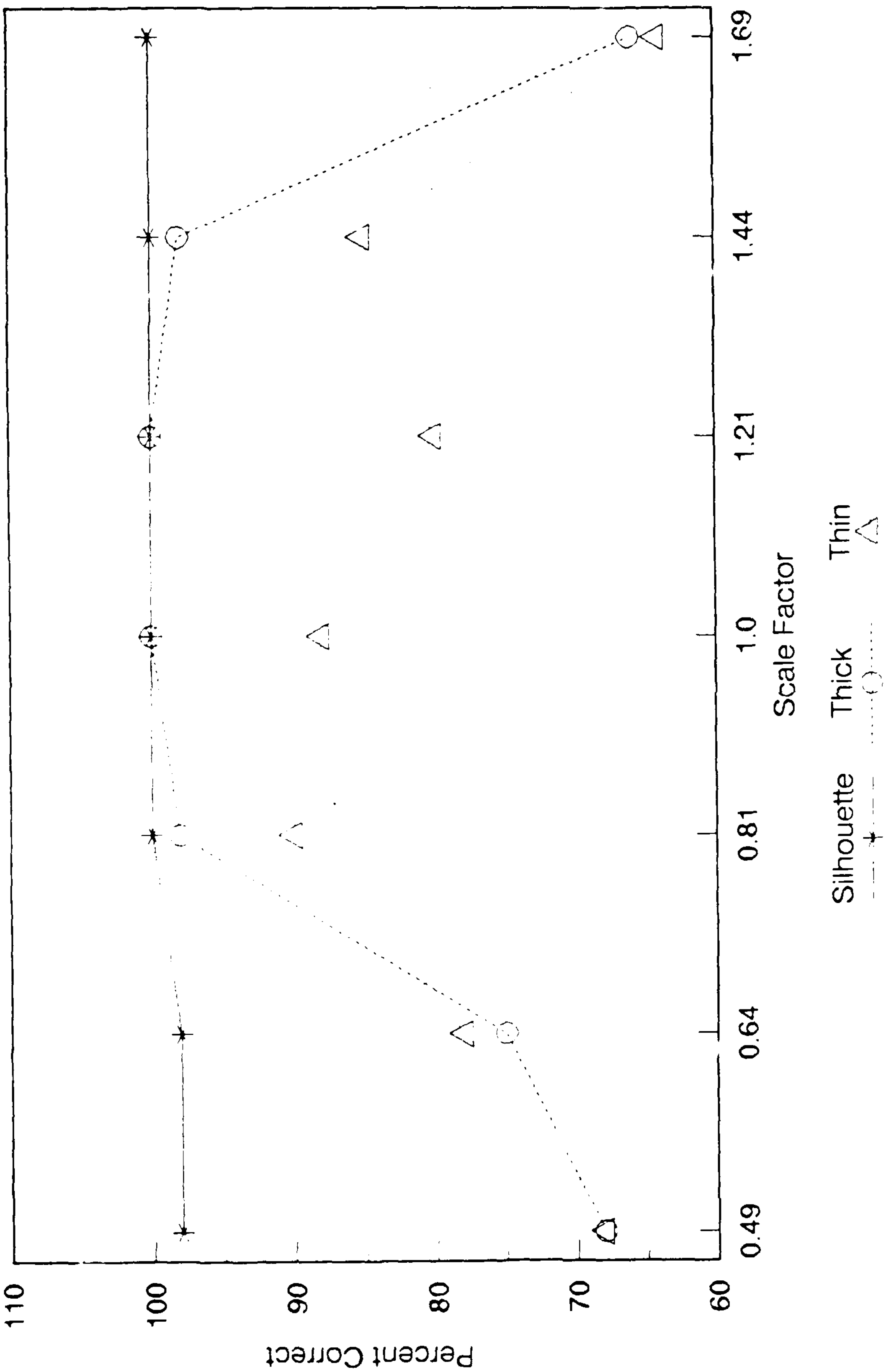


FIGURE 9. NETWORK PERFORMANCE AS A FUNCTION OF SCALE

TABLE 1. OVERALL PERFORMANCE OF THE NETWORK USING THREE TYPES OF IMAGES

IMAGE TYPE	NO. OF ITERATIONS	% CORRECT
SILHOUETTE	200	95.2
	1000	99.0
THICK EDGE	200	83.0
	300	88.0
	400	90.0
	1000	84.0
THIN EDGE	400	63.0
	600	70.0
	1000	63.0

REFERENCES

1. Lippmann, Richard P. "An Introduction to Computing with Neural Nets," IEEE ASSP, April 1987, pp. 4-22.

2. Larimore, W.E., "Statistical Inference on Stationary Random Fields," Proc. IEEE, Vol. 65, 1977, pp. 961-970.

3. Kashyap, R.L. and Chellapa R., "Estimation and Choice of Neighbors in Spatial Interaction Models of Images," IEEE Trans. Inform. Theory, Vol. 1T-29, 1983, pp. 60-72.

4. Faugers, O.D. and Toscani G., "The Calibration Problem for Stereo," Proc. CVPR, 1986, pp. 15-20.

5. Lowe, D.G., Three-Dimensional Object Recognition From Single 2-D Images, Comp. Sci. Division, New York Univ., TR-202, 1986.

6. Farsaie, A., "An Artificial Neural System For Target Identification Using IR Imagery Data," Internal Report, NAVSWC-WO, 1989.

7. Giles, C., Griffin, R., and Maxwell, T., "Encoding Geometric Invariances In Higher Order Neural Networks," Proc. IEEE, 1987.

8. Reid, M., Spirkovska, L., and Ochoa, E., "Rapid Training of Higher Order Neural Networks For Invariant Pattern Recognition," Proc. IJCNN, 1989.

9. Kohonen, T., Self-Organization and Associative Memory, Springer-Verlag Co., 1987.

10. Kohonen, T., "Improved Versions of Learning Vector Quantization," Proc. IJCNN, 1990.

DISTRIBUTION

	Copies
Defense Technical Information Center Cameron Station Alexandria, VA 22314	4
Library of Congress Attn: Gift and Exchange Division Washington, DC 20540	4
Internal Distribution	
C	1
D2	1
D4	1
E	1
E231	2
E232	3
E342 (GIDEP)	1
F	1
G	1
G06	1
G07	1
G20	1
G30	1
G40	1
G42	5
H	1
J	1
K	1
K10	1
K40	1
K44 (L. Reid)	1
K50	1
N	1
N30	1
N40	1
N41 (R. McClintock)	1
R	1
R04	1
R05	1
R40	1
R44	1
U	1
U04	1
U20	1
U30	1

REPORT DOCUMENTATION PAGE

Form Approved OMB No 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
October 1990

3. REPORT TYPE AND DATES COVERED
Final

4. TITLE AND SUBTITLE
Invariant Target Recognition Using Self-Organizing Networks

5. FUNDING NUMBERS

6. AUTHOR(S)
Beth Farrar Ali Farsaie J. Joseph Fuller

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

8. PERFORMING ORGANIZATION REPORT NUMBER
NAVSWC TR 90-418

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This paper presents an overview of neural networks technology and specifics of the self-organizing networks. An artificial neural network is described that recognizes objects regardless of their spatial orientation. This network is invariant to rotation, scale, and translation.

Invariance is built into the network by introducing a unique set of features that were developed in-house. This overcame the two shortcomings of long training times and combinatorial explosion of terms often present in other networks. Preliminary results suggests that with further refinement and enhancement, the system described in this report will have the capability to reliably recognize targets under adverse environmental conditions.

14. SUBJECT TERMS
Neural Networks Target Recognition Invariant Features
Self Organized Feature Maps

15. NUMBER OF PAGES
28

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and its title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

BLOCK 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If Known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD	- See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	- See authorities.
NASA	- See Handbook NHB 2200.2
NTIS	- Leave blank.

Block 12b. Distribution Code.

DOD	- Leave blank.
DOE	- Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA	- Leave blank.
NTIS	- Leave blank.

Block 13. Abstract. Include a brief ***(Maximum 200 words)*** factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code ***(NTIS only)***.

Blocks 17.-19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.